REMARKS

The claims were rejected on the sole reference constituting a small portion of a book by Patterson. The office action indicates that the functional units described therein are processors because a processor "is merely something that operates on data." This is not the case. A "processor" is defined in Microsoft Press Computer Dictionary, Third Edition, (copy attached as Exhibit A) as a "central processing unit, microprocessor."

The functional units described in the excerpt from the Patterson book are not central processing units. The small portion of the book describes a CDC 6600 computer. Attached, as Exhibit B, is a printout from a Web page discussing the CDC 6600 and 7600 computers. In the first block under the heading Architecture are a number of hyperlinks to information, including a 6600 system diagram. The 6600 system diagram is attached as Exhibit C. There, it can be seen that the various functional units referred to in the Patterson books are actually functional units of one central processor.

Therefore, there is no register accessible by a plurality of processors in the CDC 6600 and reconsideration is respectfully requested.

Respectfully submitted,

Date: March 7, 2005

[illegible]

Timothy N. Trop, Reg. No. 28,994
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Ste. 100
Houston, TX 77024
713/468-8880 [Phone]
713/468-8883 [Fax]

Attorneys for Intel Corporation

Microsoft Press

Computer Dictionary

Third Edition

Microsoft® Press

gramming element is the procedure (a named sequence of statements, such as a routine, subroutine, or function). The most widely used high-level languages (C, Pascal, Basic, FORTRAN, COBOL, Ada) are all procedural languages. *See also* procedure. *Compare* nonprocedural language.

procedural rendering \prə-sē`jər-əl ren´dər-ēng\ *n.* The rendering of a two-dimensional image from three-dimensional coordinates with texturing according to user-specified conditions, such as direction and degree of lighting.

procedure \prə-sē´jər\ *n.* In a program, a named sequence of statements, often with associated constants, data types, and variables, that usually performs a single task. A procedure can usually be called (executed) by other procedures, as well as by the main body of the program. Some languages distinguish between a procedure and a function, with the latter (the function) returning a value. *See also* function, parameter, procedural language, routine, subroutine.

procedure call \prə-sē´jər käl`\ *n.* In programming, an instruction that causes a procedure to be executed. A procedure call can be located in another procedure or in the main body of the program. *See also* procedure.

process[1] \pros´es\ *n.* A program or part of a program; a coherent sequence of steps undertaken by a program.

process[2] \pros´es\ *vb.* To manipulate data with a program.

process-bound \pros´es-bound`\ *adj.* Limited in performance by processing requirements. *See also* computation-bound.

process color \pros´es kəl`ər\ *n.* A method of handling color in a document in which each block of color is separated into its subtractive primary color components for printing: cyan, magenta, and yellow (as well as black). All other colors are created by blending layers of various sizes of halftone spots printed in cyan, magenta, and yellow to create the image. *See also* color model, color separation (definition 1). *Compare* spot color.

processing \pros´es-ēng\ *n.* The manipulation of data within a computer system. Processing is the vital step between receiving data (input) and producing results (output)—the task for which computers are designed.

processor \pros´es-ər\ *n. See* central processing unit, microprocessor.

Processor Direct Slot \pros`es-ər-dər-ekt` slot´\ *n.* See PDS (definition 1).

Prodigy Information Service \prod`ə-jē in-fər-mā´shən sər`vəs\ *n.* An online information service founded by IBM and Sears. Like its competitors America Online and CompuServe, Prodigy offers access to databases and file libraries, online chat, special interest groups, e-mail, and Internet connectivity. *Also called* Prodigy.

product \prod´ukt\ *n.* **1.** An operator in the relational algebra used in database management that, when applied to two existing relations (tables), results in the creation of a new table containing all possible ordered concatenations (combinations) of tuples (rows) from the first relation with tuples from the second. The number of rows in the resulting relation is the product of the number of rows in the two source relations. *Also called* Cartesian product. *Compare* inner join. **2.** In mathematics, the result of multiplying two or more numbers. **3.** In the most general sense, an entity conceived and developed for the purpose of competing in a commercial market. Although computers are products, the term is more commonly applied to software, peripherals, and accessories in the computing arena.

production system \prə-duk´shən si`stəm\ *n.* In expert systems, an approach to problem solving based on an "IF this, THEN that" approach that uses a set of rules, a database of information, and a "rule interpreter" to match premises with facts and form a conclusion. Production systems are also known as rule-based systems or inference systems. *See also* expert system.

Professional Graphics Adapter \prə-fesh`ə-nəl graf´iks ə-dap`tər\ *n.* A video adapter introduced by IBM, primarily for CAD applications. The Professional Graphics Adapter is capable of displaying 256 colors, with a horizontal resolution of 640 pixels and a vertical resolution of 480 pixels. *Acronym:* PGA (P`G-A´).

Professional Graphics Display \prə-fesh`ə-nəl graf´iks dis-plā`\ *n.* An analog display introduced by IBM, intended for use with their Professional Graphics Adapter. *See also* Professional Graphics Adapter.

***** Please note, this page (and web site) are in early development.
Items are certainly not complete, and may be inaccurate.
Your information, comments, corrections, etc. are eagerly requested.
Send e-mail to Ed Thelen. Please include the URL under discussion. Thank you *****

CDC 6600 & 7600

Manufacturer	ID	location	Date	from
Control Data Corp	CDC 6600- S/N 1	**	1964	Lawrence Livermore Laboratory
Control Data Corp	CDC 7600- S/N 1	**	1969	Lawrence Livermore Laboratory

- Also see 6600 WORD.doc and 7600 WORD.doc by Ron Mak

- Also see Dr. Dobbs Journal

Architecture

- each 60 bit word held:
  - 1 floating point value
  - 10 6 bit characters (pre ASCII :-)
  - 4 15 bit (register) instructions or fewer 30 bit (memory access) instructions
- 1 instruction could be issued each 0.1 microseconds, conditional upon functional unit, register, and data availability
- 1 instruction word **look-ahead, 12 previous instruction words instantly available**
- **Gordon Bell** lecture notes
- **Gordon Bell** 6600 system diagram **and also see the next diagram (#40)**
- **James E. Thornton "Parallel operation in the Control Data 6600", in Chapter 39 of "Computer Structures: Readings and Examples" by C. Gordon Bell & Allen Newell**

Special features - CDC 6600

- 10 independent "Functional Units" in the Main Processor included:
  - 2 floating point Multipliers (1 microsecond)
  - 1 floating point Divider (3.4 microseconds)
  - 1 Add and 1 Long Add units (0.3 microseconds)
  - 2 increment (used for memory access), 1 branch, 1 boolean, and 1 shift unit (each 0.1 microseconds)
- 10 (optionally 20) built in "Peripheral Processors" (PPs) controlled the main processor, operated the control console (including painting the alphanumerics in vector mode), and performed all I/O (Input/Output)

  12 bit (register) and 24 bit (memory address in 18 bits) instructions

  each had 4096 12 bit words, each could access any peripheral channel

- Memory bandwidth was 1 60 bit word per 100 nanoseconds (10 per

microsecond
- Memory access time 475 nanoseconds
- X shape helped reduce signal transit time. Outer parts of the **X** held not time critical things like heat exchangers, refrigerator pummps, "Dead Start" panel (most other computers were "booted" up, CDC usage was to "Dead Start" a computer, etc.
- No individual lamps and switches for machine registers, total operator access was via the two big CRT (TV tube) displays and keyboard
- Circuit cards were 3 by 3 inches, in pairs, with transistors soldered onto each circuit card, and resistors & connection wires soldered between. Called "cord wood" modules - see photo
- Meory modules were 1024 bits by 12 bits wide (no parity), (like the CDC 160A) 5 of these units made 1024 - 60 bit words.
- Every Cray machine seemed to have a "Population Count" instruction, which counted the 1 bits in a word - we figured this was for 1 customer - NSA. (National Security Agency) is charged with code cracking, and those who claim to know say the number of bits in a field is interesting.
- 18 bit addresses, word addressable - Seymore must have figured that no one would ever buy that much memory because he originally used the 18th bit for another memory function. Boeing insisted on 262,144 (256 K words) memory, and and Engineering Change Order did the re-work to allow that much
- A 2 megaword Extended Core Storage (ECS) was available. It was phased to match main memory, so after a 1.4 microsecond start up delay, 10 - 60 bit words could be written to or read from ECS.
- Assuming no register conflicts, one instruction was issued each 100 nanoseconds
- The instruction buffer was large enough to hold FFT code assuming precomputed trig values - usually done if you were going to do much FFT work.

Interesting option - from Tom Kelly

These were 6600's. SN 82 and 84, I think, but that is very hazy. The customer was the Naval Air Development Center, in Johnsville (or Warminster), PA. They had been modified for real-time use, and with special interfaces to simulation hardware. If I recall (and I didn't do a lot with this), if you referenced a negative address, it accessed the simulation hardware. There was a hardware realtime scheduler.

> I had heard of a modification for extra memory for Boeing,
> but never a "monitor mode" for the CPU.
> Why would some want to use a "monitor mode"?

It was a standard option. I've looked it up. Look at: "Control Data 6000 Series Computer Systems: Reference Manual", Pub # 60100000, Revision N, Appendix F. (1972)

If the CEJ/MEJ (Central Exchange Jump/Monitor Exchange Jump) option was installed, then there was a "monitor flag" that controlled the operation of the XJ (CPU) and MXN (PPU) instructions. The instruction lists on the covers indicate that the MXN instruction was "Included in 6700 or those systems having the applicable Standard Options" The appendix refers to this as "monitor mode" on p. F-5.

When the CPU was running the CPUMTR code, it ran with the monitor flag set, which prevented the PPs from interrupting it again.

Special features & CDC 7600

- **Quirks** - the **Scope operating system** expected a time limit **parameter** on the job control card. Oddly, you were forced to give the job time limit in **OCTAL SECONDS**. There was a limit of 5 octal digits permited in any field of the job control card, so 77777 (octal) was the maximum time you could request. (If your job took more compute time than that, your job was terminated, - output files to that point were retained.)

  Octal 77777 converts into 32767 (decimal) seconds, 546 (decimal) minutes, or **9.1 hours**.

  Several books now quote a mean-time-between-failures (MTBF) for the 6600 as 9 hours. I could not imagine where that number could have come from. (Our 6600's ran months with out unscheduled maintenance time. Every few months the CDC Customer Engineers would (rather rudely) demand machine time - like to do Engineering Change Orders). I now suspect that the quoted 9 hour MTBF in recent books came inadvertently from the maximum CPU time requestable.

  To give an idea of the reliability of the 6600, I liked to program and calculate Pi as a method of learning a new machine's assembly language. I did it to 500,000 decimal places in about 60 hours on a CDC-6600 one Thanksgiving weekend. Because of the length of time, I had to write a check point dump of the intermediate results after 8 hours, and restart the job from the check point dump, over and over until the job was done. That meant I went into work every 8 hours all during that weekend until the 60 hour running time was complete. There was no worry on my part that the 6600 would fail during that weekend run - and the value of Pi was correct as determined later from faster machines with larger memories.

- **Quirks** - the use of OCTAL parameters on the Scope job control card was passionately defended by most other Control Data employees. They could not imagine why an expensive computer should have to do a decimal to binary (octal) conversion when a human could do it instead. (Really true, Unbelivable!!)
- The **CDC 7600** was **main processor code compatable** with the CDC 6600,

  and the clock was ALMOST 4 times faster. 27.5 nanoseconds
  - Extensive **pipelining** within the functional units permitted even more

than a **4x speed increase** over the 6600.
- More peripheral units were standard, used slightly differently
- see Gordon Bell slide sequence
- different packaging, smaller, tighter
- nine functional units instead of 10 in 6600

Interesting Web Sites

-
-
-

Time Line

```
from http://ei.cs.vt.edu/~history/Parallel.html
        Control Data Corporation (CDC) founded.

from http://wotug.ukc.ac.uk/parallel/documents/misc/timeline/timeline.txt
========1960========

Control Data starts development of CDC 6600.  (MW: CDC, Cray, CDC
6600)
========1964========
Control Data Corporation produces CDC 6600, the world's first
commercial supercomputer.  (GVW: CDC, Cray, CDC 6600)

========1969========
CDC produces CDC 7600 pipelined supercomputer.  (GVW: CDC, Cray, CDC
7600)
========1972========

Seymour Cray leaves Control Data Corporation, founds Cray Research
Inc.  (GVW: CDC, CRI)
```

Number produced

50 as per http://www.newmedianews.com/tech_hist/cdc6600.html

Other information

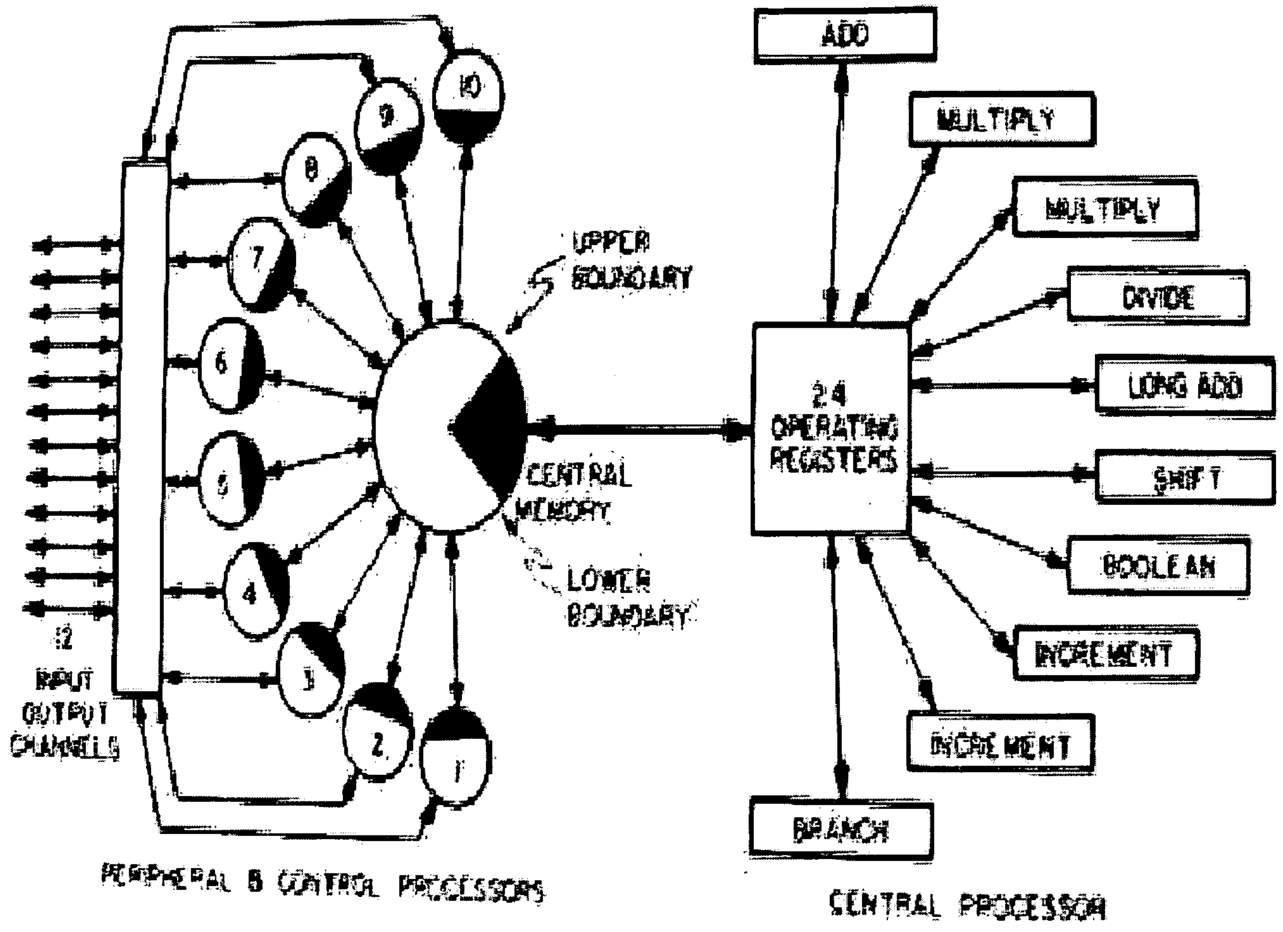
There is a popular scientific benchmark called the Linpack Benchmark used to measure the speed that a particular computer can complete a particular "compute bound" task. As per Linpack Benchmark

---

If you have comments or suggestions, Send e-mail to Ed Thelen

Go to Docent Training home page
Go to Visual Storage page
Go to top

Updated February 19, 2001



Slide 39 of 89

Notes:

The Thornton book is really great because it contains both the design philosophy and the design. It describes their quest for both speed through circuitry and parallelism. It is also a must read for every computer architect.

One principle was that all parts of the computer should operate asynchronously and independently. To begin with all IO devices could independently. Next any IO channel could be assigned to any IO device so that the channel doesn't become a bottleneck. Finally, any IO Channel could be assigned to any of the ten IO Peripheral Processing Units or PPUs. PPUs were totally independent 12 bit computers similar to the CDC 160. All PPUs could transfer data to the central memory. There was unbound flexibility for the transfer of data into the large computer's memory. Of course the central memory was accessed by the main processing unit where all the floating point computation was carried out on 60-bit words.